

1 CLAIMS

2

3 1. One or more computer readable media having stored thereon a
4 plurality of instructions that, when executed by one or more processors, causes the
5 one or more processors to perform acts including:

6 identifying a plurality of modules in a software program, wherein each
7 module includes a plurality of blocks and wherein the plurality of modules
8 includes checker modules;

9 for each of the plurality of modules,

10 generating an original checkpoint value, and

11 incorporating the original checkpoint value into a checker module;

12 and

13 for each of the checker modules,

14 generating a new checkpoint value after the original checkpoint
15 value has been incorporated into the checker module, and

16 determining a new block to add to the checker module to offset the
17 incorporated original checkpoint value such that subsequent generation of a
18 checkpoint value for the checker module equals the original checkpoint
19 value for the checker module.

20

21 2. One or more computer readable media as recited in claim 1, wherein
22 the incorporating comprises incorporating the original checkpoint value into
23 multiple checker modules.

1 3. One or more computer readable media as recited in claim 1, wherein
2 the generating comprises:

3 computing, based on the plurality of blocks of a module, a message
4 authentication code (MAC) value to be used as the checkpoint value for the
5 module.

6
7 4. One or more computer readable media as recited in claim 3, wherein
8 the computing comprises:

9 inputting each of the plurality of blocks of the module into an exclusive-or
10 operator that generates an output value by performing an exclusive-or operation on
11 each block and an encrypted version of the previous output of the exclusive-or
12 operator; and

13 using, as the message authentication code value, the output value from the
14 exclusive-or operator obtained from inputting the last of the plurality of blocks
15 into the exclusive-or operator.

16
17 5. One or more computer readable media as recited in claim 1, wherein
18 the determining a new block comprises:

19 encrypting the new checkpoint value; and

20 determining, as the content of the new block, a value equal to the exclusive-
21 or of the encrypted new checkpoint value and the original checksum value.

22
23 6. One or more computer readable media as recited in claim 1, wherein
24 the new block does not alter the functionality of the module.

1 7. One or more computer readable media as recited in claim 1, wherein
2 the new block comprises a data block.

3
4 8. One or more computer readable media as recited in claim 1, wherein
5 the plurality of instructions, when executed, further causes the one or more
6 processors to perform acts including adding, prior to generating the new
7 checkpoint value, additional instructions to the module as part of one or more
8 additional blocks, the additional instructions causing the addition of the new block
9 to not alter the functionality of the module.

10
11 9. One or more computer readable media as recited in claim 1, wherein
12 the software program further includes a plurality of checkpoints corresponding to
13 the incorporated checkpoint values, wherein each checkpoint identifies when the
14 integrity of the corresponding module is to be verified.

15
16 10. A method comprising:
17 identifying a plurality of segments in an object; and
18 applying cyclic integrity verification to the object based on the plurality of
19 segments.

20
21 11. A method as recited in claim 10, wherein the cyclic integrity
22 verification is applied to the plurality of segments by:
23 for each of the plurality of segments,
24 generating an original checkpoint value, and

1 incorporating the original checkpoint value into a checker segment;
2 and

3 for each of the checker segments,

4 generating a new checkpoint value after the original checkpoint
5 value has been incorporated into the checker segment,

6 determining an additional block to be added to the checker segment
7 to offset the incorporated original checkpoint value such that subsequent
8 generation of a checkpoint value for the checker segment equals the
9 original checkpoint value for the checker segment.

10
11 **12.** A method as recited in claim 10, wherein the cyclic integrity
12 verification is applied to verify the shape of the plurality of segments.

13
14 **13.** A method as recited in claim 10, wherein the cyclic integrity
15 verification is applied to verify the behavior of the plurality of segments.

16
17 **14.** One or more computer-readable memories comprising computer-
18 readable instructions that, when executed by a processor, direct a computer system
19 to perform the method as recited in claim 10.

20
21 **15.** A method comprising:
22 identifying a plurality of segments in an object;
23 generating a checkpoint value for each of the plurality of segments;
24 storing the checkpoint value for each of the plurality of segments in another
25 of the plurality of segments; and

1 modifying each of the plurality of segments so that the addition of the
2 checkpoint value to the segment is offset and the checkpoint value for the segment
3 remains the same.

4

5 **16.** A method as recited in claim 15, wherein the storing comprises
6 storing the checkpoint value into multiple other segments of the plurality of
7 segments.

8

9 **17.** A method as recited in claim 15, wherein the modifying comprises:
10 computing, based on a plurality of blocks of a segment, a message
11 authentication code (MAC) value to be used as the checkpoint value for the
12 segment; and

13 determining a new block to add to the segment to offset the stored
14 checkpoint value such that subsequent generation of a checkpoint value for the
15 segment equals the previously generated message authentication code value.

16

17 **18.** A method as recited in claim 17, wherein the computing comprises:
18 inputting each of the plurality of blocks of the segment into an exclusive-or
19 operator that generates an output value by performing an exclusive-or operation on
20 each block and an encrypted version of the previous output of the exclusive-or
21 operator; and

22 using, as the message authentication code value, the output value from the
23 exclusive-or operator obtained from inputting the last of the plurality of blocks
24 into the exclusive-or operator.

1 **19.** A method as recited in claim 17, wherein the determining a new
2 block comprises:

3 generating a new checkpoint value based on the plurality of blocks and
4 including the stored checkpoint value;

5 encrypting the new checkpoint value; and

6 determining, as the content of the new block, a value equal to the exclusive-
7 or of the encrypted new checkpoint value and the original checksum value.

8
9 **20.** A method as recited in claim 15, wherein the modifying does not
10 alter the functionality of the segment.

11
12 **21.** A method as recited in claim 15, wherein the modifying comprises
13 adding a new data block.

14
15 **22.** A method as recited in claim 15, wherein the object comprises a
16 software program.

17
18 **23.** A method as recited in claim 15, further comprising storing a
19 checkpoint corresponding to each checkpoint value, each checkpoint identifying
20 when the integrity of the corresponding segment is to be verified.

21
22 **24.** One or more computer-readable memories comprising computer-
23 readable instructions that, when executed by a processor, direct a computer system
24 to perform the method as recited in claim 15.

1 **25.** A method comprising:

2 generating a verification value for a first segment of an object;

3 generating an original verification value for a second segment of the object;

4 adding, to the second segment, the verification value for the first segment;

5 and

6 adding an offset value to the second segment so that a newly calculated
7 verification value for the second segment equals the original verification value.

8

9 **26.** A method as recited in claim 25, wherein the generating the
10 verification value for the first segment comprises generating the verification value
11 based at least in part on behavior of the first segment during execution of the first
12 segment.

13

14 **27.** A method as recited in claim 26, wherein the behavior of the first
15 segment during execution includes modification of a register by one or more
16 instructions in the first segment during execution.

17

18 **28.** A method as recited in claim 25, further comprising:

19 adding, to the first segment, the original verification value for the second
20 segment; and

21 adding another offset value to the first segment so that a newly calculated
22 verification value for the first segment equals the verification value for the first
23 segment.

1 **29.** A method as recited in claim 25, wherein the generating the original
2 verification value comprises:

3 computing, based on a plurality of blocks of the second segment, a message
4 authentication code (MAC) value.

5
6 **30.** A method as recited in claim 29, wherein the computing comprises:

7 inputting each of the plurality of blocks of the second segment into an
8 exclusive-or operator that generates an output value by performing an exclusive-or
9 operation on each block and an encrypted version of the previous output of the
10 exclusive-or operator; and

11 using, as the message authentication code value, the output value from the
12 exclusive-or operator obtained from inputting the last of the plurality of blocks
13 into the exclusive-or operator.

14
15 **31.** A method as recited in claim 25, wherein the adding an offset value
16 comprises:

17 generating a new verification value for the second segment;
18 encrypting the new verification value; and
19 determining, as the offset value, a value equal to the exclusive-or of the
20 encrypted new verification value and the original verification value.

21
22 **32.** A method as recited in claim 25, wherein the offset value does not
23 alter the functionality of the module.

1 **33.** A method as recited in claim 25, wherein the offset value comprises
2 a data block.
3

4 **34.** A method as recited in claim 25, wherein the object comprises a
5 software program.
6

7 **35.** A method as recited in claim 25, further comprising storing a
8 checkpoint, corresponding to the verification value, that identifies when the
9 integrity of the first segment is to be verified.
10

11 **36.** A method as recited in claim 35, further comprising storing the
12 checkpoint in the second segment.
13

14 **37.** One or more computer-readable memories comprising computer-
15 readable instructions that, when executed by a processor, direct a computer system
16 to perform the method as recited in claim 25.
17

18 **38.** One or more computer-readable media having stored thereon a
19 computer program including:
20

21 a plurality of segments, each including one or more checkpoint values to be
used to verify the integrity of one or more other segments; and
22

23 wherein the plurality of segments further include a plurality of checkpoints
that identify a circular ordering of verifying the integrity of the segments.
24

1 **39.** One or more computer-readable media as recited in claim 38,
2 wherein each of the checkpoint values is message authentication code (MAC)
3 value based on the one or more other segments.

4
5 **40.** One or more computer-readable media as recited in claim 38,
6 wherein each of the plurality of segments includes a checkpoint value to be used to
7 verify the integrity of each of the other of the plurality of segments.

8
9 **41.** A production system, comprising:
10 a memory to store an original program; and
11 a production server equipped with a cyclic integrity verification protection
12 tool that is used to augment the original program for protection purposes, the
13 production server being configured to parse the original program into a plurality of
14 segments and apply cyclic integrity verification to the plurality of segments.

15
16 **42.** A production system as recited in claim 41, wherein the cyclic
17 integrity verification is applied to the plurality of segments by:

18 for each of the plurality of segments,

19 generating an original checkpoint value, and

20 incorporating the original checkpoint value into a checker segment;

21 and

22 for each of the checker segments,

23 generating a new checkpoint value after the original checkpoint
24 value has been incorporated into the checker segment, and

1 determining an additional block to be added to the checker segment
2 to offset the incorporated original checkpoint value such that subsequent
3 generation of a checkpoint value for the checker segment equals the
4 original checkpoint value for the checker segment.

5

6 **43.** A production system as recited in claim 42, wherein the cyclic
7 integrity verification is applied to the plurality of segments by further including a
8 plurality of checkpoints corresponding to the incorporated checkpoint values,
9 wherein each checkpoint identifies when the integrity of the corresponding
10 segment is to be verified.

11

12 **44.** A client-server system, comprising:
13 a production server to apply cyclic integrity verification to a program to
14 produce a protected; and
15 a client to store and execute the protected program, the client being
16 configured to evaluate the protected program to determine whether the protected
17 program has been tampered with.